CE/CZ4034 Information Retrieval

AY22/23 S2 - Group 10

Crypto.io -

A NFT-related Information Retrieval Project

Assignment Report

ISABELLE ONG EE LING (U1921109A)

JOLENE TAN (U1921255B)

TAN YAP SIANG (U1920756H)

TAN ZHI WEI SAMUEL (U2020677G)

TEO GUANG XIANG (U2020948F)

List Of Tables

List Of Figures

Table Of Contents

1   Introduction

1.1 Background Information

In recent years, NFTs (Non-Fungible Tokens) have gained immense popularity as a new form of digital asset ownership and collection. NFTs represent unique digital items such as art, music, videos, and more, which are verified on a blockchain and cannot be replicated or exchanged for something else. NFTs have been around since 2014 when they were introduced on the Ethereum blockchain, but did not become popular until the second half of 2020. This was brought about due to the COVID-19 pandemic and subsequent lockdowns around the world, which led to an increase in online activity, and the need for digital ownership and authenticity became more apparent.

However, as with any emerging technology or trend, there are varying opinions and sentiments about NFTs amongst the general public. Public sentiment can have a significant effect on the NFT market as they are still a relatively new concept, and their value is largely based on community perception and demand. If the public sentiment towards NFTs is positive, an increase in demand is typically observed, which can drive up prices and create a bullish market. However, the converse can be observed if the public sentiment turns negative, where there is decrease in demand which can lead to lower prices and a bearish market.

1.2 Project Objective and Scope

This project aims to provide a comprehensive analysis of the NFT market, by analyzing the sentiment of text data related to 10 selected NFT collections, including positive, negative, and neutral opinions. The goal in doing so is to provide insights into the overall sentiment of the NFT community. Text data will be collected from Twitter, which is a social media platform with a rich source of real-time, publicly available data with diverse opinions. Relevant information can be retrieved from the data collected when users submit queries in the form of keywords or phrases to the search engine built. A time series analysis will also be performed to identify market

trends and patterns. This project hopes to not only provide users with a better understanding of the driving sentiment of the NFT market, but also allow users to monitor the sentiment of the community, helping them make more informed investment and business decisions.

1.3   Team Structure

Name	Contributions
ISABELLE ONG EE LING	- Data Exploration - Classification - Report - Slides
JOLENE TAN	- Crawling - Classification - User Interface - Report - Slides
TAN YAP SIANG	- Crawling - Classification - User Interface - Report - Slides
TAN ZHI WEI SAMUEL	- Indexing - User Interface - Report - Slides
TEO GUANG XIANG	- Data Exploration - Classification - Report - Slides

Table 1. Team Structure

1.4   Selection of NFTs

After conducting research on numerous NFT collections, the following 10 marketplaces, as shown in Table 2 below, have been selected for analysis in this

project. The rationale behind these selections include how popular or unpopular they are, as well as the availability of tweets related to them.

NFT	Release Date
Mutant Ape Yacht Club	August 2021
Azuki	January 2022
Bored Ape Yacht Club	April 2021
CloneX	November 2021
Meebits	November 2022
The Potatoz	July 2022
CryptoPunks	June 2017
Phanta Bear	January 2022
MekaVerse	October 2021
Pixelmon	February 2022

Table 2. NFT Collections

2 Crawling

There are 2 popular libraries to scrape tweets from Twitter, snscrape and tweepy. In this project, snscrape is chosen. The main reason behind this choice is that snscrape allows access to historical tweets whereas tweepy only allows access to tweets from the past 7 days. As time series analysis of the NFTs is a goal of this project, analyzing historical data is essential. Moreover, snscrape does not rely on the Twitter API, unlike tweepy, and therefore does not have rate limits or restrictions. As this project requires a large number of tweets (minimum 10,000) to be scrapped, this was also an important consideration. Snscrape also has more flexible search parameters than tweepy, allowing for a greater diversity of tweets to be collected.

It is to be noted that in the query, retweets and replies are being excluded. The rationale behind this is that the dataset should reflect as accurately as possible, the

sentiment of the general public toward the NFT collection, in order for a useful analysis to be made. Retweets and replies undermine this as retweets are duplicate records which could result in a biased dataset, while replies usually lack contextual information which could result in the dataset not being insightful.

Table 3 below shows the information that is being crawled from the tweets.

Field	Description
data	Date tweet was created
id	ID of tweet
rawContent	Text content of tweet
user.username	Username of user who created tweet
replyCount	Count of replies
retweetCount	Count of retweets
likeCount	Count of likes

Table 3. Tweet Fields

For each NFT collection, about 2000 tweets are scraped. As one of the project objectives is to perform time series analysis on each marketplace, it was ensured that an equal number of tweets from each quarter, starting from 2021, is collected.

In addition, the release date of the collection was also taken into account. For instance, Pixelmon was released in February 2022. This means that there are 5 relevant quarters to scrape tweets from, January 2022 to March 2022, April 2022 to June 2022, July 2022 to September 2022, October 2022 to December 2022 and January 2023 to March 2023. 500 tweets will be scraped from each of these quarters. Table 4 below shows the number of Pixelmon-related tweets collected from each quarter. It can be seen that the number of tweets collected from each quarter is almost equal, which will enhance the reliability of the time series analysis. Note that data collection is being done in February, explaining the lack of tweets in the quarter

of January 2023 to March 2023 as only 2 months of data are available compared to the rest with 3 months of available data.

Quarter	Number of Tweets
January 2022 to March 2022	449
April 2022 to June 2022	452
July 2022 to September 2022	347
October 2022 to December 2022	467
January 2023 to March 2023	112

Table 4. Quarter Counts of Pixelmon Tweets

3 Dataset Creation

3.1 Full Dataset

Before concatenation of the 10 individual NFT datasets collected, a new column indicating the name of the specific NFT collection is appended to each dataset. For instance, 'Pixelmon' will be the value in the 'NFT' column for tweets related to Pixelmon that were scraped.

3.2   Labeled Dataset

Approximately 10% (2000 tweets) of the full data will be extracted for manual labeling by the team members. In order to ensure each NFT collection is fairly represented, approximately 10% (200 tweets) of tweets from each marketplace will be extracted. In addition, to ensure that not all tweets come from a certain quarter, the full dataset will be shuffled before the extraction is done.

The columns 'Polarity (Reviewer 1)' and 'Polarity (Reviewer 2)' are then appended to the dataset to be labeled. All 5 team members will be taking part in the creation of the labeled dataset, by first manually labeling 400 tweets each as 'Reviewer 1', and then another 400 tweets each as 'Reviewer 2'. Labels are standardized to 'pos' for subjective positive-sentiment tweets, 'neg' for subjective negative-sentiment tweets and 'neu' for objective tweets.

In order to give the team members an insight on what the possible sentiment of the tweet might be, unsupervised learning sentiment analysis tools Vader and Textblob are used. Vader is a sentiment analysis tool that is part of NLTK, a popular Python library for natural language processing. It is a rule-based sentiment analysis tool that is designed to analyze the sentiment of text data and provide a score that indicates the level of positivity, negativity, or neutrality expressed in the text. It works by using a combination of lexicons and grammatical rules to analyze the text and determine the sentiment expressed in it. Textblob, on the other hand, is a Python library for performing sentiment analysis. It uses a rule-based approach, which involves analyzing the text using a predefined set of rules and heuristics to determine the sentiment of the text. The sentiment of the tweet generated by Vader and Textblob will be appended to the dataset as new columns, 'vader' and 'textblob' respectively.

In the event that there are conflicting sentiment labels between reviewer 1 and 2, the 2 reviewers will look through the tweet again and come to a conclusion on its final sentiment label together.

4   Data Exploration

4.1   Number of records and words

Table 5 reflects the number of records, words and distinct words in the crawled dataset.

	Full Dataset
Number of records	20295
Number of words	338601
Number of distinct words	20315

Table 5. Full Dataset Corpus Information

4.2 Unique Words for each Sentiment Label

Table 6 and 7 reflects the number of words, distinct words and unique words in the augmented train and test dataset respectively. Note that in this case, distinct words are defined as the total number of different words in the sentiment label, while unique words are defined as words that can only be found in that specific sentiment label.

	Positive	**Negative**	**Neutral**
Number of words	16241	12627	6914
Number of distinct words	2706	1587	1939
Number of unique words	1641	973	994

Table 6. Augmented Train Dataset Corpus Information

	Positive	**Negative**	**Neutral**
Number of words	4264	390	1731

Number of distinct words	1189	260	681
Number of unique words	866	128	377

Table 7. Test Dataset Corpus Information

4.3 Visualizations

As part of our data exploration process, a word cloud representation is used to visualize the most frequently occurring words in each sentiment label of the augmented train dataset, as shown in Figures 1, 3 and 5. The size of each word cloud indicates its frequency, with larger words appearing more frequently in the data. In addition, the top 20 most frequent tokens for each sentiment label are also identified to gain a better understanding of relevant tokens in each sentiment label, as shown in Figures 2, 4 and 6.

Figure 1. WordCloud Visualization of Positive Sentiment

Figure 2. Top 20 Frequent Tokens of Positive Sentiment

Figure 3. WordCloud Visualization of Negative Sentiment

Figure 4. Top 20 Frequent Tokens of Negative Sentiment

Figure 5. WordCloud Visualization of Neutral Sentiment

Figure 6. Top 20 Frequent Tokens of Neutral Sentiment

From Figures 3 and 4, we can observe that there are words such as 'malware', 'rug' and 'steal' that are common in the negative sentiment. This gives us insights on the context of the dataset where NFT users are wary of the risks such as a potential rug-pull, NFT thefts and malware attacks.

5   Data Preprocessing

Special preprocessing is performed due to the "noisy" nature of text data from Twitter. Most of the tweets contain URLs, hashtags and mentions which often do not add value to sentiment analysis. In order to transform the tweets into a standardized form for later tasks, removal of non-English and duplicated tweets, stopword removal, tokenization, noise removal, case folding, and lemmatization are performed.

5.1   Techniques

5.1.1 Removal of Non-English and Duplicated Tweets

As English is chosen as the main communication language, non-English tweets were removed. Duplicated tweets are also removed as multiple copies of the same data artificially inflate the size of the dataset, which could lead to overfitting and skewed models.

5.1.2 Stopword Removal and Tokenization

Tokenization involves breaking down a tweet into smaller units or tokens, making it easier to analyze and understand their meanings. Additionally, stopword removal is carried out, using NLTK, to eliminate common words that hold little semantic value and thus do not contribute much to the sentimental analysis of tweets.

5.1.3 Noise Removal

Due to the "noisy" nature of tweets, noise removal is performed to remove the following:

- URLs
- Hashtags (#) and mentions (@)
- Non-UTF-8 or ASCII characters
- Line feed (\r) and carriage return (\n)

5.1.4 Case Folding and Lemmatization

To reduce all alphabets to lowercase, case folding is performed. Even though uppercase letters could display the emotions of users, such as being agitated and excited, the terms alone are believed to sufficiently convey their subjectivity and polarity. As such, case folding is conducted.

NLTK WordNet Lemmatizer is used to reduce words to its lemma form, for easier analysis and comparison. If a word cannot be located in Wordnet, it will be returned in its original form without any changes.

5.2 Evaluation

In order to evaluate the effectiveness of the preprocessing techniques utilized, the training time, accuracy and macro average f1-score using a baseline XGBoost classifier model is considered. Table 8 below shows the classification report before and after preprocessing techniques were being applied.

Metrics	Before Preprocessing	After Preprocessing
Time	9.47s	1.73s
Accuracy	0.82	0.83
Precision	0.75	0.73
Recall	0.63	0.65
F1-score	0.64	0.67

Table 8. Classification Report Before and After Preprocessing

From the results obtained, it can be seen that training time, accuracy, as well as f1-score all improved after the application of the preprocessing techniques. This indicates the effectiveness of the preprocessing techniques selected and hence, will continue to be used throughout in the following sentiment analysis classification tasks.

6 Data Augmentation

6.1 Rationale

Attempts were made to find a public sentiment analysis labeled dataset to be used as training data for the classification models. However, it was eventually concluded that there was absence of one that accurately encapsulates the contextual-specifications of the dataset used in this project. Hence, it was decided that the manually labeled dataset will be used to train and evaluate the various classification models.

However, it was discovered that the dataset was highly skewed in nature, which could lead to several issues. Table 9 below shows the value counts of each sentiment in the dataset. It can be observed that the number of positive tweets far outnumber the number of negative tweets (almost 13 times more). The highly imbalanced nature of the data may lead to the classification model being biased towards the majority class, leading to inaccurate results and thus misclassification of the minority class. The model may not have enough examples of the minority class to learn meaningful patterns and relationships, making it less useful in practice.

Sentiment	Count
pos	1374
neu	517
neg	109

Table 9. Labeled Data Sentiment Counts

6.1.1. Random Undersampling

Random undersampling of the majority class is one of the solutions considered to solve this problem. Random undersampling is a technique used to balance imbalanced datasets. This involves randomly selecting a subset of samples from the majority class so that the number of samples in the majority class is equal to the number of samples in the minority class, thus helping to balance the dataset and prevent the model from being biased towards the majority class. Although random undersampling can help to improve the performance of the classification model, it can also lead to a loss of information as some samples from the majority class may be discarded. This is especially true in the context of this project where there is an extreme class imbalance where hundreds of positive and neutral tweets need to be discarded for this solution to be utilized.

6.1.2. Data Augmentation

Data augmentation is another solution that was considered to solve this problem. Data augmentation is a technique used to increase the amount of training data by creating new synthetic data points based on the existing data. This can be

particularly effective in solving imbalance datasets because it allows for the creation of additional samples for the minority class without collecting new data. By generating synthetic data for the minority class, the dataset is more balanced and the classification model has more examples to learn from, thus increasing the overall performance and accuracy. Hence, this will be the chosen solution to tackle the problem of having a highly skewed dataset in this project.

It is to be noted that only the train data will be augmented and the augmented data will not be used for the final time series analysis as these are not actual tweets that represent the general public sentiment toward a particular NFT collection.

6.2 Experiments

6.2.1 Overview

There are a variety of data augmentation techniques for textual data. In order to select the best one to use for this particular dataset, several experiments with different augmentors are being conducted. In order to evaluate the effectiveness of the data augmentation techniques utilized, accuracy, f1-scores for each class as well as the macro average f1-score using a baseline XGBoost classifier model is considered. Note that only negative tweets will be augmented due to the extreme imbalance between this class and the other 2 classes.

6.2.2 Baseline

The labeled dataset will be split into train and test data using a 80-20 train-test split. As data augmentation will only be done on the train data, the sentiment counts of the train data is shown in Table 10 below.

Sentiment	Count
pos	1097
neu	415
neg	88

Table 10. Train Data Sentiment Counts Before Augmentation

6.2.3 Synonym Augmentor

In the first experiment, the `SynonymAug` provided by the `nlpaug` package will be utilized. This augments text data at the sentence level by replacing words with their synonyms. When applied to a sentence, it randomly selects words from the sentence and replaces them with one of their synonyms. The synonyms are obtained using word embeddings, which are a way of representing words as vectors in a high-dimensional space. In order to control the level of augmentation applied to the data, several parameters can be adjusted. Table 11 below shows the parameters specified for use in this project.

Parameter	Description	Value
aug_src	Specifies the source of the data to be augmented	wordnet
aug_max	Maximum number of words to replace in each sentence	3
aug_n	Number of generated augmented sentences for each original sentence	3

Table 11. SynonymAug Parameters

The sentiment counts of the train data after augmentation is shown in Table 12 below.

Sentiment	Count
pos	1097
neu	415
neg	352

Table 12. Train Data Counts After Synonym Augmentation

6.2.4 Embedding Augmentor

In the next experiment, the `EmbeddingAugmenter` provided by the `textattack` package will be utilized. This is used to generate perturbations to the input text by

replacing words with their nearest neighbors in an embedding space. It uses pre-trained word embeddings to compute the distance between words and find their nearest neighbors. Although various parameters can be adjusted to control the behavior of this augmentor, their default values will be utilized for this project. Table 13 below shows the values of some notable parameters provided by this augmentor.

Parameter	Description	Value
model_type	Pre-trained word embedding model to use	glove
top_n	Number of nearest neighbors to consider when searching for replacements	50
distance_metric	Distance metric to use when computing distances between words	cosine

Table 13. EmbeddingAugmenter Parameters

The sentiment counts of the train data after augmentation is shown in Table 14 below.

Sentiment	Count
pos	1097
neu	415
neg	176

Table 14. Train Data Counts After Embedding Augmentation

6.2.5 Stacking of Synonym and Embedding Augmentors

In the first 2 experiments, 2 individual augmentors are being experimented with. However, due to the extreme imbalance nature of the dataset, it is seen that the counts obtained after the augmentations are still far from ideal. In addition, the classification results indicate that after these augmentation techniques were applied, the model performs worse than before. This led to the idea of stacking the 2 augmentors, not only to increase the number of negative tweets, but also to increase the diversity and effectiveness of the generated data. The rationale behind doing so is so that a wider variety of permutations to the input text can be generated, thus

exposing the classification model to a wider range of variations in the input date, in hopes of improving its ability to generalize to new data. In addition, different augmentors have different sets of strengths and weaknesses and are better suited to different types of data. By combining them, the strengths of each augmentor can be leveraged, thus mitigating their weaknesses and resulting in more effective data augmentation.

As the augmentors will be stacked sequentially, experiments will be done to determine what is the ideal order of stacking them. This experiment first applies the synonym augmentor followed by the embedding augmentor.

The sentiment counts of the train data after each round of augmentation is shown in Table 15 below.

Sentiment	(1st) Synonym	(2nd) Embedding
pos	1097	1097
neu	415	415
neg	352	704

Table 15. Train Data Counts After Synonym + Embedding Augmentation

6.2.6 Stacking of Embedding and Synonym Augmentors

This experiment first applies the embedding augmentor followed by the synonym augmentor.

The sentiment counts of the train data after each round of augmentation is shown in Table 16 below.

Sentiment	(1st) Embedding	(2nd) Synonym
pos	1097	1097
neu	415	415
neg	176	704

Table 16. Train Data Counts After Embedding + Synonym Augmentation

6.3 Evaluation

Table 17 provides an overview of the results of the data augmentation experiments performed.

	Baseline	Synonym	Embedding	Synonym + Embedding	Embedding + Synonym
Count (neg)	88	352	176	704	704
Accuracy	0.83	0.81	0.82	0.82	0.80
Precision (neg)	0.64	0.53	0.57	0.58	0.53
Precision (pos)	0.91	0.89	0.89	0.89	0.89
Precision (neu)	0.68	0.66	0.67	0.67	0.64
Precision	0.74	0.69	0.71	0.71	0.68
Recall (neg)	0.43	0.43	0.38	0.52	0.48
Recall (pos)	0.89	0.88	0.88	0.89	0.87
Recall (neu)	0.76	0.71	0.75	0.70	0.69
Recall	0.69	0.67	0.67	0.70	0.68
F1-score (neg)	0.51	0.47	0.46	0.55	0.50
F1-score (pos)	0.90	0.89	0.89	0.89	0.88

F1-score (neu)	0.72	0.68	0.70	0.68	0.66
F1-score	0.71	0.68	0.68	0.71	0.68

Table 17. Data Augmentation Experiment Results

From the results obtained, it can be observed that the individual augmentors are not effective and the classification model actually performs worse than when it is trained on the baseline data. This suggests that the method of augmentation used by these augmentors are not effective in the specific use case of this project.

In addition, the number of negative tweets generated are not significant enough to solve the imbalance issue. This is resolved by sequentially stacking the augmentors, resulting in a lot more negative tweets being generated.

It is also observed that applying the Synonym augmentor followed by the Embedding augmentor improved the performance of the classification model from the baseline. The f1-score for the negative class improves rather significantly while not resulting in a major trade-off between accuracy and the f1-scores of the positive and neutral classes. Thus, the sequential stacking of the Synonym augmentor and Embedding augmentor will be the chosen data augmentation approach to be used. A set of training data with the sentiment counts as seen in Table 15 is generated and will be used throughout in the following sentiment analysis classification tasks.

It is to be noted that the same training data will be used to train the models instead of generating new data or splitting the data in a different manner every time. The rationale behind doing so is not only to provide a fair comparison of the performance of the different models, but also prevent unfortunate splits scenarios.

7 Indexing

7.1 Elasticsearch

Indexing is done in order to facilitate efficient and effective search and retrieval. To achieve this, Elasticsearch is utilized. Elasticsearch is a highly scalable open-source

search engine with full-text search and real-time analytics capabilities. It is built on top of the Apache Lucene search library and uses Lucene's indexing capabilities to efficiently index large volumes of data.

7.2　Indexing in Elasticsearch

Elasticsearch indexes data by storing it in an inverted index, which is a critical component of any search engine.

When data is ingested into Elasticsearch, it is first tokenized, which breaks it down into smaller units called tokens. Depending on the analyzer, tokenization divides the data into individual words, phrases, or other units. The analyzer is a programmable module that controls how data is tokenized and processed, including stop words removal, stemming, and lowercasing.

The analyzer's tokens are then appended to the inverted index. The inverted index is a mapping of terms or tokens to the documents and positions within those documents that they correspond to. The mapping used in this project is as shown in Figure 7 below. Each term is associated with a list of documents that contain that term, as well as the positions in each document where the term appears. Descriptions of the information retrieved are given in Table 18 below.

Elasticsearch divides the inverted index into smaller, more manageable chunks using a data structure known as a shard. Each shard is an independent index that can be stored on a different node in a distributed Elasticsearch cluster. Elasticsearch can now scale horizontally by adding more nodes to the cluster, allowing it to handle larger volumes of data and higher query loads.

```
mapping = {
    "mappings": {
        "properties": {
            "Datetime": {"type": "date"},
            "Likes": {"type": "integer"},
            "NFT": {"type": "keyword"},
            "Text": {"type": "text"},
            "Clean": {"type": "text"},
            "Polarity": {"type": "text"}
        }
    }
}
```

Field	Description	Example
Datetime	Timestamp of the tweet	"02-03-2022 2.31"
Likes	Number of Likes	12
NFT	NFT name	"Phanta Bear"
Clean Text	Tweet text that has been pre-cleaned	"Azuki Best"
Polarity	Values representing polarity. 1 is positive, 0 is neutral and -1 is negative	"0.491320649"

Table 18. Information Retrieved

7.3   Querying in Elasticsearch

In Elasticsearch, querying is done using the Elasticsearch Query DSL (Domain Specific Language), which is a powerful and flexible query language that allows users to construct complex queries for searching and filtering data. After the query is constructed by the user, it is sent to the Elasticsearch cluster, which processes the query and returns a ranked list of documents that match it. Elasticsearch provides a wide range of querying parameters that can be used to customize and refine search results.

In this project, users have the option to do either a traditional search or a multifaceted search. For traditional search , the user enters a query, and the search engine returns a list of results that match the query. The parameters used are as shown in Figure 8 with descriptions of them provided in Table 19.

```
parameters = {"query": {
                "bool": {
                    "should": [
                        {
                            "match": {
                                "NFT": {
                                    "query": query,
                                    "boost": 10,
                                    "fuzziness": 2,
                                }
                            }
                        },
                        {
                            "match": {
                                "Clean": {
                                    "query": query,
                                    "fuzziness": 2,
                                    "boost": 1
                                }
                            }
                        },
                    ]
                }
            }
```

Figure 8. Traditional Search Querying Parameters

In addition, in order to help users narrow down the search results quickly and efficiently, without having to browse through a large number of irrelevant results, as well as discover new information that they may not have considered before, multifaceted search is also implemented. This technique allows users to filter search results using multiple criteria simultaneously. The results are presented as a set of categories or facets that allow the user to refine the search results based on various dimensions. The parameters used are as shown in Figure 9 with descriptions of them provided in Table 19.

```
parameters = {"query": {
        "bool": {
            "must": [{
                    "query_string": {
                    "query": searchNFT
                    }
                }
            ],
            "should": [
                {
                    "match": {
                        "NFT": {
                            "query": query,
                            "boost": 10,
                            "fuzziness": 2,
                        }
                    }
                },
                {
                    "match": {
                        "Clean": {
                            "query": query,
                            "fuzziness": 2,
                            "boost": 1
                        }
                    }
                },
            ]
        }
    }
```

Figure 9. Multifaceted Search Querying Parameters

Parameter	Description
Should	At least one of these clauses must match, like logical OR. Because Elasticsearch returns a result where each tuple has an attached score, we can actually take all the results and return the top K.
Must	Specifies that matching documents must satisfy specified conditions for the clause to be considered a match.
Match	Specifies the condition in which the should clause will accept. For our dataset we are matching queries on both the NFT name and Clean Text Column (Cleaning done prior).
Fuzziness	Specifies the maximum edit distance between the query and our

	matching condition
Boost	Puts more weight on the query, we are boosting the NFT name to be weighted more so that if the NFT name is included somewhere in the query we'll be displaying results related to that.

Table 19. Description of Querying Parameters

7.4.2 Evaluation

In order to evaluate the quality of querying, 5 queries will be written, with their results shown in Figures 10, 11, 12, 13, 14. The speed of querying is also shown in Table 20.

```
2023-04-07 14:50:25.614 POST http://localhost:9200/ir_assignment_try/_search [status:200 duration:0.028s]
                 Datetime  Likes                                                   Text Polarity
NFT
Azuki  2022-06-28T00:00:00    124  holding 2 Azuki and 4 Beanz, why do I think @A...          1
Azuki  2022-12-30T00:00:00    119  No matter what you say @AzukiOfficial is the c...          1
Azuki  2022-03-30T00:00:00      2  Best #NFT to buy according to @B2Community\n\n...          1
Azuki  2022-12-30T00:00:00      0  Between Dec. 26 and 27, Azuki was the best-per...          1
Azuki  2022-03-30T00:00:00      7  Fading Azuki red beans because I was married t...         -1
Azuki  2023-02-13T00:00:00      0  Both cool! Azuki NFT are the coolest!\n#Azuki\...          1
Azuki  2022-09-28T00:00:00      0  The best part about being in azuki is that the...          1
Azuki  2022-09-28T00:00:00      6  The best part about being in azuki is that the...          1
Azuki  2022-06-28T00:00:00      4  Which of these memes said it best? #degods #ba...          1
Azuki  2022-12-30T00:00:00      0  Bean #13475 swapped on NFT Trader #BEANZ #Azuk...          1
[]
```

Figure 10. "azuki best" + NFT: Azuki Query Results

```
2023-04-06 16:33:03.301 POST http://localhost:9200/ir_assignment4/_search [status:200 duration:0.018s]
                       Datetime  Likes                                             Text Polarity
NFT
Bored Ape Yacht Club  2021-09-18T00:00:00      5  Looking to buy another NFT - I need good proje...          1
CloneX                2022-06-27T00:00:00    173  In the NFT space we are the people who make th...         -1
CloneX                2022-06-23T00:00:00    103  My Top 5 teams for Avatar NFT projects..\n\n1....          1
CloneX                2022-09-24T00:00:00      2  CloneX TW is so connected and open-minded to c...          1
CloneX                2023-10-02T00:00:00      1  What's going on with CloneX good entry?\n#NFTC...          1
Bored Ape Yacht Club  2021-06-19T00:00:00    211  Bored Ape Yacht Club has successfully transiti...          1
CloneX                2022-09-12T00:00:00      0  Oh my God ! so cool i can not wait to get mine...          1
CloneX                2022-12-24T00:00:00      1  TOP 5 NFT Projects Update &amp; NFT News | Coo...          1
CloneX                2022-09-20T00:00:00     99  Felt like I needed to diversify a bit in this ...          1
CloneX                2023-06-02T00:00:00      2  Congratulations @Khalissman on flipping CloneX...          1
[]
```

Figure 11. "nft good projects" + NFT: Bored Ape Yacht Club + NFT: CloneX" Query Results

```
2023-04-07 15:04:29.035 POST http://localhost:9200/ir_assignment_try/_search [status:200 duration:0.028s]
                  Datetime  Likes                                             Text Polarity
NFT
Phanta Bear  2022-11-01T00:00:00      0  台湾のアイコンJayChouのPhantaBearNFTがBoredApe YachtClu...      0
Phanta Bear  2022-01-13T00:00:00      1  Phanta Bear 暴跌暴漲關鍵五分鐘 — 瘋狂回顧/心得 \nhttps://t.co...      0
Phanta Bear  2022-04-02T00:00:00      0  Woohoo!!! I am now a Phanta Bear owner!! Jay C...      1
[]
```

Figure 12. "Jaychou" + 3 search results Query Results

```
2023-04-07 15:05:23.060 POST http://localhost:9200/ir_assignment_try/_search [status:200 duration:0.008s]
              Datetime  Likes                                             Text Polarity
NFT
Meebits  2023-02-13T00:00:00      0  Meebit #12900 bought for 3.09 WETH (4,573.57 U...      1
Meebits  2023-02-13T00:00:00      0  Meebit #18044 bought for 2.88 ETH (4,245.36 US...      1
Meebits  2023-02-13T00:00:00      1  Meebit #3408 bought for 2.90 WETH (4,305.12 US...      1
Meebits  2023-02-13T00:00:00      1  Meebit #12957 bought for 4.00 ETH (5,924.68 US...      1
Meebits  2023-02-13T00:00:00      0  Meebits Top5 Sales within 24H [ 07:01, Feb 13,...      1
Meebits  2023-02-13T00:00:00    140  These are my current bags:\n1) 1x MAYC\n2) 27x...      1
Meebits  2023-02-13T00:00:00      4  Meebits triple floor check! \n#OpenSea listing...      0
Meebits  2023-02-13T00:00:00     50  Web 3 and NFT change our life\nNice to met Ape...      1
Meebits  2023-02-13T00:00:00      2  🔥🔥🔥 Hot free mint alert 🔥🔥🔥\n\nProject: NFTBox...      1
Meebits  2023-02-13T00:00:00      0  🐋 Top Whale SELLS 🔻 (Last hour):\n\nAnimetas: 1...      1
[]
```

Figure 13. " " + NFT: Meebits Query Results

```
2023-04-07 15:06:54.047 POST http://localhost:9200/ir_assignment_try/_search [status:200 duration:0.013s]
              Datetime  Likes                                             Text Polarity
NFT
Meebits  2022-12-21T00:00:00     14  My Top 3 NFT picks:\n\n1. CryptoPunks\n2. Auto...      1
Meebits  2023-01-28T00:00:00     56  4 of the top 6 most profitable NFT trader wall...      1
Meebits  2022-12-30T00:00:00      4  Here, Top 10 NFT sales of 2022\n(expect Yugala...      1
Meebits  2022-12-12T00:00:00      6  NFT market Highlights:\nCollections like Meebi...      1
Meebits  2023-04-02T00:00:00      0  Yuga Labs Obtains Cryptopunks and Meebits Bran...     -1
Meebits  2023-04-02T00:00:00      0  Yuga Labs Obtains Cryptopunks and Meebits Bran...     -1
Meebits  2023-01-22T00:00:00      9  NFT LIST 🔑🧵\n#MutantApeYachtClub\n#BoredApeYach...      0
Meebits  2023-01-19T00:00:00      5  🚨#BREAKING: #Mailchimp has🍌announced that it w...     -1
Meebits  2022-12-13T00:00:00     21  Our guys can't wait to explore the @OthersideM...      0
Meebits  2023-01-02T00:00:00      2  #YugaLabs have tweeted that they have faced so...     -1
[]
```

Figure 14. 'cryptopunk buy' Query Results

Query	Time Taken (s)
"azuki best" + NFT: Azuki	0.028
"nft good projects" + NFT: Bored Ape Yacht Club + NFT: CloneX	0.018
"Jaychou" + 3 search results	0.028

“ ” + NFT: Meebits	0.008
“Cryptopunk” + NFT: Meebits	0.013

Table 20. Query Performance

8 Classification

8.1 Methodology

The pipeline for sentiment analysis can be seen from Figure 15 below. The details of each step will be provided in the following sections.

Figure 15. Sentiment Analysis Pipeline

8.2 Evaluation Metrics

Table 21 below shows the metrics that are being used to evaluate the performance of the classification models. It is noted that accuracy is not the only evaluation metrics that is being considered here as although a high model accuracy is desirable, it is largely influenced by class imbalance, which was an identified problem of the dataset being used in this project.

In addition, due to the skewed nature of the dataset used in this project, 2 variations of precision, recall and f1-score will be considered. The first being the precision, recall and f1-score for each class which allows insights on how well the model is performing for each sentiment class to be gained, as well as identify areas for improvement. The second is the macro average precision, recall and f1-score which takes the average of the F1-scores of each class, providing an overall measure of the model's performance that is not biased towards any particular class. This allows insights on how well the model is performing overall to be gained, as well as identify areas for improvement that may be masked by class imbalance.

It is noted that because positive and negative sentiment tweets help to draw more useful insights and because f1-score calculations take into account precision and recall, accuracy, f1-scores for each sentiment class and the macro-average f1-scores will be the most important evaluation metrics considered to evaluate the classification models.

Metrics	Description
Training Time	Amount of time it takes to train a classification model on a dataset of labeled texts in order to predict the sentiment of new, unseen texts
Accuracy	Percentage of correctly predicted sentiment labels out of all instances in a test dataset
Precision for each class	Precision calculated separately for each sentiment class
Macro Average Precision	Proportion of correctly predicted positive neutral or negative sentiment instances among all instances predicted as positive or neutral or negative by model
Recall for each class	Recall calculated separately for each sentiment class
Macro Average Recall	Proportion of correctly predicted positive neutral or negative sentiment instances among all actual positive neutral or negative instances in dataset
F1-scores for each class	F1-score calculated separately for each sentiment class
Macro Average F1-score	Harmonic mean of precision and recall, with higher values indicating better model performance

Table 21. Evaluation Metrics

8.3   Unsupervised Learning

It was previously mentioned that unsupervised learning tools Vader and Textblob were being utilized to give annotators an insight on what the possible sentiment of the tweet might be. These tools are then evaluated after manual labeling was completed.

8.3.1 Vader

Table 22 below shows the classification report of the Vader classification model.

Metrics	Value
Accuracy	0.44
Precision (neg)	0.53
Precision (pos)	0.42
Precision (neu)	0.49
Precision	0.48
Recall (neg)	0.27
Recall (pos)	0.70
Recall (neu)	0.26
Recall	0.41
F1-score (neg)	0.36
F1-score (pos)	0.52
F1-score (neu)	0.34
F1-score	0.41

Table 22. Classification Report of Vader model

8.3.2 TextBlob

Table 23 below shows the classification report of the Textblob classification model.

Metrics	Value
Accuracy	0.37
Precision (neg)	0.41
Precision (pos)	0.35
Precision (neu)	0.42
Precision	0.39
Recall (neg)	0.14
Recall (pos)	0.69
Recall (neu)	0.22
Recall	0.35
F1-score (neg)	0.21
F1-score (pos)	0.47
F1-score (neu)	0.29
F1-score	0.32

Table 23. Classification Report of Textblob Model

8.3.3 Evaluation

From the results obtained, it is observed that both tools do not yield good results, with accuracies less than 50% and f1-scores less than 0.5. This could be due to the complexity of the language in this dataset. As data was obtained from social media (Twitter), there is a high chance it contains a high degree of sarcasm and irony.

In addition, unsupervised learning tools do not take into account the context and tone of the text, which can significantly affect the sentiment, as the same words can have

different sentiment depending on the context which the tools are not able to differentiate between. Due to the poor performance of these classifiers, they will not be considered as classification models to be applied on the full dataset.

8.4 Machine Learning

8.4.1 Feature Extraction

This step refers to the process of transforming raw text data into numerical features that can be used as input to a machine learning model. The goal is to represent the text data as a numerical vector that captures the important information about the text, such as the frequency of certain words or phrases. This allows machine learning algorithms to learn patterns and relationships between the features and the sentiment labels.

The Term Frequency-Inverse Document Frequency (TF-IDF) feature extraction method will be used in this case. This method evaluates the importance of a word in a document, relative to its frequency in a corpus and can be used to identify the most important words in a piece of text that are indicative of a particular sentiment, by assigning a weight to each word in the text based on its frequency and inverse frequency in the corpus.

8.4.2 Hyperparameter Tuning Approach

Where applicable, the hyperparameters of the baseline classification models will be tuned, with the objective of selecting the optimal values for the model hyperparameters, in order to improve the performance of the model.

In this project, the grid search approach will be used with scoring done using 5-fold cross-validation and macro average f1-scores. This involves identifying a set of hyperparameters and their corresponding values, before the model is trained and evaluated for each combination of hyperparameters. Due to the exhaustive nature of this search approach, it can be computationally expensive for large parameter spaces. In order to mitigate this, several rounds of hyperparameter tuning will be done for each model, with each round only aiming to find the optimal values of a single parameter. The most important parameters will be tuned first before moving

on to tune other less critical parameters, allowing the model to be more refined and focused. This also prevents combinatorial explosion in the number of possible combinations, which makes it difficult to explore the entire hyperparameter space and find the optimal combination.

8.4.3 XGBoost

8.4.3.1　　Description

XGBoost is a popular gradient boosting machine learning algorithm that can be used for classification tasks such as, in this case, sentiment analysis. The XGBoost algorithm works by iteratively adding decision trees to a model, with each subsequent tree aiming to correct the errors of the previous tree. The algorithm optimizes a loss function to find the best split at each node of the tree, and uses a regularization term to prevent overfitting. The resulting model is a combination of multiple decision trees, which can be used to predict the sentiment of new texts.

8.4.3.2　　Baseline Model

The `XGBClassifier` from the `xgboost` package will be used. Figure 16 below shows the parameters provided by the baseline model.

```
{'objective': 'binary:logistic',
 'use_label_encoder': None,
 'base_score': None,
 'booster': None,
 'callbacks': None,
 'colsample_bylevel': None,
 'colsample_bynode': None,
 'colsample_bytree': None,
 'early_stopping_rounds': None,
 'enable_categorical': False,
 'eval_metric': None,
 'feature_types': None,
 'gamma': None,
 'gpu_id': None,
 'grow_policy': None,
 'importance_type': None,
 'interaction_constraints': None,
 'learning_rate': None,
 'max_bin': None,
 'max_cat_threshold': None,
 'max_cat_to_onehot': None,
 'max_delta_step': None,
 'max_depth': None,
 'max_leaves': None,
 'min_child_weight': None,
 'missing': nan,
 'monotone_constraints': None,
 'n_estimators': 100,
 'n_jobs': None,
 'num_parallel_tree': None,
 'predictor': None,
 'random_state': None,
 'reg_alpha': None,
 'reg_lambda': None,
 'sampling_method': None,
 'scale_pos_weight': None,
 'subsample': None,
 'tree_method': None,
 'validate_parameters': None,
 'verbosity': None}
```

Figure 16. Baseline XGBoost Parameters

8.4.3.3 Hyperparameter Tuning

Table 24 below specifies the hyperparameters that are tuned, as well as the order of tuning them, for the XGBoost classification model.

Hyperparameter	Description

learning_rate	(Tuned first) Controls how quickly the model adapts to the training data
max_depth	Maximum depth of each tree in the ensemble
n_estimators	Number of trees in the ensemble
min_child_weight	Minimum sum of instance weight needed in a child
subsample	Fraction of training instances used to train each tree
colsample_bytree	(Tuned last) Fraction of features used to train each tree

Table 24. XGBoost Hyperparameter Descriptions

Table 25 below shows the range of values considered for the tuning of each hyperparameter, as well as the optimal values found for each of them.

Hyperparameter	Value Range	Optimal Value
learning_rate	[0.3, 0.1, 0.01]	0.3
max_depth	[4, 6, 8, 10]	6
n_estimators	[50, 100, 200, 300, 400, 500]	300
min_child_weight	[1, 3, 5, 7, 9]	1
subsample	[0.5, 0.7, 0.9, 1]	1
colsample_bytree	[0.5, 0.7, 0.9, 1]	0.9

Table 25. XGBoost Hyperparameter Values

8.4.3.4 Tuned Model

Figure 17 below shows the parameters provided by the tuned XGBoost classification model.

```
{'objective': 'multi:softprob',
 'use_label_encoder': None,
 'base_score': None,
 'booster': None,
 'callbacks': None,
 'colsample_bylevel': None,
 'colsample_bynode': None,
 'colsample_bytree': 0.9,
 'early_stopping_rounds': None,
 'enable_categorical': False,
 'eval_metric': None,
 'feature_types': None,
 'gamma': None,
 'gpu_id': None,
 'grow_policy': None,
 'importance_type': None,
 'interaction_constraints': None,
 'learning_rate': 0.3,
 'max_bin': None,
 'max_cat_threshold': None,
 'max_cat_to_onehot': None,
 'max_delta_step': None,
 'max_depth': 6,
 'max_leaves': None,
 'min_child_weight': 1,
 'missing': nan,
 'monotone_constraints': None,
 'n_estimators': 300,
 'n_jobs': None,
 'num_parallel_tree': None,
 'predictor': None,
 'random_state': None,
 'reg_alpha': None,
 'reg_lambda': None,
 'sampling_method': None,
 'scale_pos_weight': None,
 'subsample': 1,
 'tree_method': None,
 'validate_parameters': None,
 'verbosity': None}
```

Figure 17. Tuned XGBoost Parameters

8.4.3.5   Evaluation

Table 26 below provides an overview of the results of the baseline and tuned XGBoost classification models performed.

	Baseline	Tuned
Training Time	2.09s	7.03s
Accuracy	0.82	0.80

Precision (neg)	0.52	0.45
Precision (pos)	0.91	0.89
Precision (neu)	0.66	0.65
Precision	0.70	0.66
Recall (neg)	0.52	0.48
Recall (pos)	0.88	0.88
Recall (neu)	0.74	0.66
Recall	0.71	0.67
F1-score (neg)	0.52	0.47
F1-score (pos)	0.90	0.88
F1-score (neu)	0.70	0.65
F1-score	0.71	0.67

Table 26. Classification Report of XGBoost Models

From the results obtained, it is observed that the tuned model actually performs worse than the baseline model. This could be due to overfitting where the hyperparameters have been tuned to the training set so closely that the model overfits on the training data, and is thus not able to generalize well to new data. This could also be due to the lack of data that is not able to support the improved model's complexity, resulting in a decline in performance after hyperparameter tuning is conducted. The baseline XGBoost classification model will thus be chosen to be applied on the full dataset.

8.4.4 LightGBM

8.4.4.1 Description

LightGBM (Light Gradient Boosting Machine) is a popular open-source gradient boosting framework that can be used for classification tasks such as, in this case,

sentiment analysis. LightGBM is based on gradient boosting decision tree algorithms that works by iteratively adding weak learners to the model to improve its performance.

8.4.4.2 Baseline Model

The `LGBMClassifier` from the `lightgbm` package will be used. Figure 18 below shows the parameters provided by the baseline model.

```
{'boosting_type': 'gbdt',
 'class_weight': None,
 'colsample_bytree': 1.0,
 'importance_type': 'split',
 'learning_rate': 0.1,
 'max_depth': -1,
 'min_child_samples': 20,
 'min_child_weight': 0.001,
 'min_split_gain': 0.0,
 'n_estimators': 100,
 'n_jobs': -1,
 'num_leaves': 31,
 'objective': None,
 'random_state': None,
 'reg_alpha': 0.0,
 'reg_lambda': 0.0,
 'silent': 'warn',
 'subsample': 1.0,
 'subsample_for_bin': 200000,
 'subsample_freq': 0}
```

Figure 18. Baseline LightGBM Parameters

8.4.4.3 Hyperparameter Tuning

Table 27 below specifies the hyperparameters that are tuned, as well as the order of tuning them, for the LightGBM classification model.

Hyperparameter	Description
learning_rate	(Tuned first) Controls the step size during gradient descent
max_depth	Controls the maximum depth of each decision tree

num_leaves	Controls the maximum number of leaves in each decision tree
feature_fraction	Controls the fraction of features to consider at each split
bagging_fraction	Controls the fraction of data to use for each iteration
min_child_samples	(Tuned last) Controls the minimum number of samples required to form a new leaf

Table 27. LightGBM Hyperparameter Descriptions

Table 28 below shows the range of values considered for the tuning of each hyperparameter, as well as the optimal values found for each of them.

Hyperparameter	Value Range	Optimal Value
learning_rate	[0.3, 0.1, 0.01, 0.001]	0.1
max_depth	[4, 6, 8, 10]	10
num_leaves	[11, 31, 51, 71, 91]	51
feature_fraction	[0.2, 0.4, 0.6, 0.8, 1]	1
bagging_fraction	[0.2, 0.4, 0.6, 0.8, 1]	0.2
min_child_samples	[10, 20, 30, 40, 50]	10

Table 28. LightGBM Hyperparameter Values

8.4.4.4　　Tuned Model

Figure 19 below shows the parameters provided by the tuned LightGBM classification model.

```
{'boosting_type': 'gbdt',
 'class_weight': None,
 'colsample_bytree': 1.0,
 'importance_type': 'split',
 'learning_rate': 0.1,
 'max_depth': 10,
 'min_child_samples': 10,
 'min_child_weight': 0.001,
 'min_split_gain': 0.0,
 'n_estimators': 100,
 'n_jobs': -1,
 'num_leaves': 51,
 'objective': None,
 'random_state': None,
 'reg_alpha': 0.0,
 'reg_lambda': 0.0,
 'silent': 'warn',
 'subsample': 1.0,
 'subsample_for_bin': 200000,
 'subsample_freq': 0,
 'feature_fraction': 1,
 'bagging_fraction': 0.2}
```

Figure 19. Tuned LightGBM Parameters

8.4.4.5 Evaluation

Table 29 below provides an overview of the results of the baseline and tuned LightGBM classification models performed.

	Baseline	Tuned
Training Time	12.53s	0.76s
Accuracy	0.80	0.82
Precision (neg)	0.50	0.61
Precision (pos)	0.88	0.90
Precision (neu)	0.65	0.68
Precision	0.68	0.73

Recall (neg)	0.48	0.52
Recall (pos)	0.86	0.88
Recall (neu)	0.70	0.74
Recall	0.68	0.71
F1-score (neg)	0.49	0.56
F1-score (pos)	0.87	0.89
F1-score (neu)	0.67	0.71
F1-score	0.68	0.72

Table 29. Classification Report of LightGBM Models

From the results obtained, it is observed that the tuned model performs better than the baseline model in all aspects, which indicates the effectiveness of the hyperparameter tuning. The model performance improved on new, unseen data which implies it has a better generalization performance. In addition, the reduction in training time indicates that the tuning of hyperparameters helped to reduce the time and computational resources required for training the model. The tuned LightGBM classification model will thus be chosen to be applied on the full dataset.

8.4.5 K-Nearest Neighbor

8.4.5.1 Description

K-nearest neighbor (KNN) is an algorithm that can be used for classification tasks such as, in this case, sentiment analysis. KNN works by finding the K closest neighbors to a new input data point and assigning it the most common class label among those neighbors. In other words, KNN relies on the assumption that data points with similar feature values tend to belong to the same class.

8.4.5.2 Baseline Model

The `KNeighborsClassifier` from the `sklearn` package will be used. Figure 20 below shows the parameters provided by the baseline model.

```
{'algorithm': 'auto',
 'leaf_size': 30,
 'metric': 'minkowski',
 'metric_params': None,
 'n_jobs': None,
 'n_neighbors': 5,
 'p': 2,
 'weights': 'uniform'}
```

Figure 20. Baseline KNN Parameters

8.4.5.3   Hyperparameter Tuning

Table 30 below specifies the hyperparameters that are tuned, as well as the order of tuning them, for the KNN classification model.

Hyperparameter	Description
n_neighbors	(Tuned first) Number of neighbors to use for classification
weights	Weight function used in prediction
metric	(Tuned last) Distance metric used to compute distances between points

Table 30. KNN Hyperparameter Descriptions

Table 31 below shows the range of values considered for the tuning of each hyperparameter, as well as the optimal values found for each of them.

Hyperparameter	Value Range	Optimal Value
n_neighbors	[5, 10, 20, 30, 40, 50]	5
weights	["uniform", "distance"]	distance
metric	['euclidean', 'manhattan', 'minkowski']	euclidean

Table 31. KNN Hyperparameter Values

8.4.5.4   Tuned Model

Figure 21 below shows the parameters provided by the tuned KNN classification model.

```
{'algorithm': 'auto',
 'leaf_size': 30,
 'metric': 'euclidean',
 'metric_params': None,
 'n_jobs': None,
 'n_neighbors': 5,
 'p': 2,
 'weights': 'distance'}
```

Figure 21. Tuned KNN Parameters

8.4.5.5   Evaluation

Table 32 below provides an overview of the results of the baseline and tuned KNN classification models performed.

	Baseline	Tuned
Training Time	0.01s	0.02s
Accuracy	0.73	0.76
Precision (neg)	0.24	0.29
Precision (pos)	0.60	0.62
Precision (neu)	0.85	0.86
Precision	0.56	0.59
Recall (neg)	0.43	0.38
Recall (pos)	0.56	0.36
Recall (neu)	0.82	0.83
Recall	0.60	0.61

F1-score (neg)	0.31	0.33
F1-score (pos)	0.83	0.85
F1-score (neu)	0.58	0.62
F1-score	0.57	0.60

Table 32. Classification Report of KNN Models

From the results obtained, it is observed that the tuned model performs better than the baseline model in all aspects, which indicates the effectiveness of the hyperparameter tuning. The justification is similar to that provided in Section ??, besides the point about training time improvement. However, this is not important for the case of both models, as both values are so small that it can be considered negligible. The tuned KNN classification model will thus be chosen to be applied on the full dataset.

8.4.6 Support Vector Machine

8.4.6.1 Description

Support Vector Machine (SVM) is a popular machine learning algorithm that can be used for classification tasks such as, in this case, sentiment analysis. The goal of SVM is to find a hyperplane that separates the different classes in the training data with the maximum margin, based on their features.

8.4.6.2 Baseline Model

The `svm.SVC()` from the `sklearn` package will be used. Figure 22 below shows the parameters provided by the baseline model.

```
{'C': 1.0,
 'break_ties': False,
 'cache_size': 200,
 'class_weight': None,
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 3,
 'gamma': 'scale',
 'kernel': 'rbf',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}
```

Figure 22. Baseline SVM Parameters

8.4.6.3   Hyperparameter Tuning

Table 33 below specifies the hyperparameters that are tuned, as well as the order of tuning them, for the SVM classification model.

Hyperparameter	Description
C	(Tuned first) Regularization parameter that controls the trade-off between achieving a low training error and a low testing error
kernel	Kernel function used to transform the input data into a higher-dimensional space
gamma	Parameter for non-linear hyperplanes
degree	(Tuned last) Degree of the polynomial kernel function ('poly') and is ignored by all other kernels
class_weight	Can be used to balance the number of samples in each class which can be useful when the data is imbalanced

Table 33. SVM Hyperparameter Descriptions

Table 34 below shows the range of values considered for the tuning of each hyperparameter, as well as the optimal values found for each of them.

Hyperparameter	Value Range	Optimal Value
C	[0.1, 1, 10, 30, 50, 70, 100]	10
kernel	['rbf', 'linear', 'poly', 'sigmoid']	linear
gamma	[0.0001, 0.001, 0.01, 0.1, 1, 10]	0.0001
degree	[1, 2, 3, 4, 5]	1
class_weight	[None, 'balanced']	balanced

Table 34. SVM Hyperparameter Values

8.4.6.4 Tuned Model

Figure 23 below shows the parameters provided by the tuned SVM classification model.

```
{'C': 10,
 'break_ties': False,
 'cache_size': 200,
 'class_weight': 'balanced',
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 1,
 'gamma': 0.0001,
 'kernel': 'linear',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}
```

Figure 23. Tuned SVM Parameters

8.4.6.5 Evaluation

Table 35 below provides an overview of the results of the baseline and tuned SVM classification models performed.

	Baseline	Tuned
Training Time	1.23s	1.51s
Accuracy	0.82	0.81

Precision (neg)	0.65	0.50
Precision (pos)	0.85	0.88
Precision (neu)	0.74	0.66
Precision	0.75	0.68
Recall (neg)	0.62	0.43
Recall (pos)	0.92	0.88
Recall (neu)	0.58	0.67
Recall	0.71	0.66
F1-score (neg)	0.63	0.46
F1-score (pos)	0.88	0.88
F1-score (neu)	0.65	0.66
F1-score	0.72	0.67

Table 35. Classification Report of SVM Models

From the results obtained, it is observed that the tuned model actually performs worse than the baseline model. The justification is similar to that provided in Section 8.4.3.5. The baseline SVM classification model will thus be chosen to be applied on the full dataset.

8.4.7 Decision Tree

8.4.7.1 Description

Decision Tree is a popular machine learning algorithm that can be used for classification tasks such as, in this case, sentiment analysis. A decision tree is a tree-like model and it works by making a series of decisions based on the features of the input data to arrive at a final classification decision.

8.4.7.2 Baseline Model

The `DecisionTreeClassifier` from the `sklearn` package will be used. Figure 24 below shows the parameters provided by the baseline model.

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

Figure 24. Baseline Decision Tree Parameters

8.4.7.3 Hyperparameter Tuning

Table 36 below specifies the hyperparameters that are tuned, as well as the order of tuning them, for the Decision Tree classification model.

Hyperparameter	Description
max_depth	(Tuned first) Maximum depth of the decision tree which controls the complexity of the model
min_samples_split	Minimum number of samples required to split an internal node
min_samples_leaf	Minimum number of samples required to be at a leaf node
criterion	(Tuned last) Function used to measure the quality of a split

Table 36. Decision Tree Hyperparameter Descriptions

Table 37 below shows the range of values considered for the tuning of each hyperparameter, as well as the optimal values found for each of them.

Hyperparameter	Value Range	Optimal Value
max_depth	[1, 3, 5, 7, 9]	9

min_samples_split	[2, 4, 6, 8, 10]	10
min_samples_leaf	[1, 2, 3, 4, 5]	1
criterion	['gini', 'entropy']	gini

Table 37. Decision Tree Hyperparameter Values

8.4.7.4 Tuned Model

Figure 25 below shows the parameters provided by the tuned Decision Tree classification model.

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': 9,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 10,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

Figure 25. Tuned Decision Tree Parameters

8.4.7.5 Evaluation

Table 38 below provides an overview of the results of the baseline and tuned Decision Tree classification models performed.

	Baseline	Tuned
Training Time	0.57s	0.05s
Accuracy	0.76	0.76
Precision (neg)	0.35	0.54
Precision (pos)	0.86	0.79
Precision (neu)	0.61	0.67
Precision	0.60	0.67

Recall (neg)	0.38	0.62
Recall (pos)	0.83	0.92
Recall (neu)	0.66	0.35
Recall	0.62	0.63
F1-score (neg)	0.36	0.58
F1-score (pos)	0.84	0.85
F1-score (neu)	0.63	0.46
F1-score	0.61	0.63

Table 38. Classification Report of Decision Tree Models

From the results obtained, it is observed that the baseline model and tuned model has about the same performance. Although there is a significant improvement in the training times as well as the tuned models f1-score for the negative class as compared to that in the baseline model, the trade-off is that there is a significant drop in the tuned models f1-score for the neutral class as compared to that in the baseline model. Based on the fact that negative sentiment provides more insights in the context of this project, the f1-scores for the negative class is prioritized over that for the neutral class. The tuned Decision Tree classification model will thus be chosen to be applied on the full dataset.

8.4.8 Naive Bayes

8.4.8.1   Description

Naive Bayes is a popular machine learning algorithm that can be used for classification tasks such as, in this case, sentiment analysis. Naive Bayes is a probabilistic algorithm that works by calculating the probabilities of each class based on the features of the data and then selecting the class with the highest probability as the predicted class.

As the `sklearn` package provides a variety of naive bayes implementations which differ in the probability distribution adopted, instead of hyperparameter tuning a baseline naive bayes classification model, experiments will be done to select the best algorithm variation.

8.4.8.2 Multinomial Naive Bayes Model

The Multinomial Naive Bayes (MultinomialNB) classifier assumes that the frequency of each word in the text follows a multinomial distribution, and calculates the probability of each sentiment label given the frequency of each word in the text. The `MultinomialNB` from the `sklearn` package will be used.

8.4.8.3 Complement Naive Bayes Model

The Complement Naive Bayes (ComplementNB) classifier is designed to handle imbalanced datasets, where one class has significantly more examples than the others. The ComplementNB classifier adjusts the probabilities of the features for each class using the complement of the class frequency, making it more robust to imbalanced datasets. ComplementNB estimates the prior probabilities of each sentiment label and the conditional probabilities of each word given each sentiment label using the training data. The `ComplementNB` from the `sklearn` package will be used.

8.4.8.4 Bernoulli Naive Bayes Model

The Bernoulli Naive Bayes (BernoulliNB) classifier assumes that the features contribute independently to the classification. The `BernoulliNB` from the `sklearn` package will be used.

8.4.8.5 Evaluation

Table 39 below provides an overview of the results of how the different variations of naive bayes classification models performed.

	MultinomialNB	ComplementNB	BernoulliNB
Training Time	0.01s	0.01s	0.01s
Accuracy	0.79	0.78	0.78

Precision (neg)	0.37	0.29	0.45
Precision (pos)	0.84	0.91	0.82
Precision (neu)	0.87	0.73	0.73
Precision	0.69	0.64	0.67
Recall (neg)	0.81	0.76	0.67
Recall (pos)	0.94	0.84	0.94
Recall (neu)	0.39	0.63	0.37
Recall	0.71	0.74	0.66
F1-score (neg)	0.51	0.42	0.54
F1-score (pos)	0.89	0.87	0.88
F1-score (neu)	0.54	0.67	0.49
F1-score	0.65	0.66	0.64

Table 39. Classification Report of Naive Bayes models

From the results obtained, it is observed that the MultinomialNB model performs better than the ComplementNB and BernoulliNB in most of the critical aspects, namely accuracy and f1-scores for the positive and negative class. This indicates that MultinomialNB is better suited for text data. Text data typically has a high dimensionality and sparse features which is a good fit for the MultinomialNB which assumes that the features are generated from a multinomial distribution and models the likelihood of each feature given each class. This is because it is common for the same word to appear multiple times in a document. The Multinomial Naive Bayes classification model will thus be chosen to be applied on the full dataset.

8.4.9 Ensemble

An ensemble model is one that combines multiple individual classifiers to make a final prediction. By combining the predictions of multiple classifiers, the risk of bias or

overfitting, that may arise from using a single classifier, is reduced. This combines the strengths of multiple individual classifiers in an attempt to provide a more accurate and robust model. In this project, the majority voting classifier ensemble learning algorithm is used, which works by assigning a class label to a given input based on the majority vote of the individual classifiers. Two attempts were made, one involving all the machine learning models and another involving all but the worst performing, KNN, machine learning models. A subset of the classification report is shown in Table 40 below.

	Training Time	Accuracy	F1-score (neg)	F1-score (pos)	F1-score (neu)	F1-score
XGBoost	2.09s	0.82	0.52	0.90	0.70	0.71
LightGBM	0.76s	0.82	0.56	0.89	0.71	0.72
KNN	0.02s	0.76	0.33	0.85	0.62	0.60
SVM	1.23s	0.82	0.63	0.88	0.65	0.72
Decision Tree	0.05s	0.76	0.58	0.85	0.46	0.63
Naive Bayes	0.01s	0.79	0.51	0.89	0.54	0.65
Ensemble - all	-	0.82	0.59	0.89	0.66	0.71
Ensemble - w/o KNN	-	0.82	0.64	0.89	0.64	0.72

Table 40. Classification Report of Ensemble Models

From the results obtained, it is observed that the ensemble model including all but the worst performing KNN machine learning models, outperforms all the individual models. This suggests the effectiveness of the voting classifier in balancing out biases and providing a more accurate prediction. The ensemble without KNN classification model will thus be chosen to be applied on the full dataset.

8.5 Deep Learning

8.5.1 Word Embeddings (Word2Vec)

Word2Vec is an algorithm that employs a 2-layer neural network to process a text corpus and generate a set of vectors. In contrast to Tf-Idf, which is a statistical measure that generates a single vector for each document, Word2Vec generates a feature vector for each word in the document and additional processing is required to convert these vectors into a single vector or other format. Furthermore, while Tf-Idf does not account for the context of words in the corpus, Word2Vec does.

In Word2Vec, it is assumed that words with similar contexts will have similar meanings and thus similar vector representations. The similarity between two words is calculated using their cosine similarity value, and words with similar values are often grouped together in the vector space. For example, the words "collection" and "nftcollector" have related meanings and are therefore grouped together.

Advantages:

1. Intuitive algorithm that transforms raw text into labeled data by mapping target words to their context words.

2. Implicitly embeds sub-linear relationships into the vector space of words, allowing relationships to be inferred by word vectors.

Disadvantages:

1. Word2Vec cannot understand unknown or out-of vocabulary words

8.5.2 Long Short-Term Memory

8.5.2.1   LSTM

Long Short-Term Memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data.

Our team decided to implement the LSTM model using Word2Vec for the classification. The word-embedding technique is commonly used when performing sentiment analysis with a neural network. Firstly, we split the NFT tweets into tokens using the word_tokenizer from NLTK library, the tokens are then used to form the embedding matrix that allows the model to capture relationships between words and represent them in a multi-dimensional space. To ensure that the text-sequences are uniform, we also implement padding to equalize all the text sequences. As we are classifying the tweets into positive, negative or neutral sentiment, a softmax activation function is chosen, along with a categorical cross entropy. Figure 26 below shows the network architecture of the LSTM classification model.


```
Layer (type)                Output Shape
=================================================
embedding (Embedding)       (None, 265, 300)

dropout (Dropout)           (None, 265, 300)

lstm (LSTM)                 (None, 128)

dense (Dense)               (None, 3)
```

Figure 26. : LSTM Architecture

8.5.2.2   LSTM with Class Weights

Class Weights are used to address the class imbalance in the training data. Due to the class imbalance, it is possible for the model to be biased towards the majority class and perform poorly on the minority class.

Class weights assign a weight to each class that is inversely proportional to its frequency in the training data. This means that the minority class will have a higher weight than the majority class. During training, the model will penalize mistakes on the minority class more heavily than the mistakes on the majority class.

8.5.2.3   Evaluation

Table 41 below provides an overview of the results of the LSTM and the LSTM with class weights classification models performed.

	LSTM	LSTM + Class Weights

Training Time	38.70s	114.10s
Accuracy	0.81	0.79
Precision (neg)	0.55	0.57
Precision (pos)	0.90	0.90
Precision (neu)	0.63	0.60
Precision	0.69	0.69
Recall (neg)	0.52	0.38
Recall (pos)	0.86	0.83
Recall (neu)	0.73	0.76
Recall	0.70	0.66
F1-score (neg)	0.54	0.46
F1-score (pos)	0.88	0.87
F1-score (neu)	0.67	0.67
F1-score	0.70	0.67

Table 41. Classification Report of LSTM Models

From the results obtained, it is observed that the LSTM with class weights model actually performs worse than the LSTM model. A possible reason for this is that using class weights may result in the model being overly biased towards the minority class, leading to worse overall performance. This could happen if the minority class is not actually more difficult to classify than the majority class, which highlights the effectiveness of the data augmentation techniques utilized to generate more data for the minority negative class. In this case, the LSTM model without class weights may be better able to learn the underlying patterns in the data and make better predictions overall because it is already designed to handle sequential data and can learn to prioritize certain patterns based on their importance in predicting the target

variable. The LSTM classification model will thus be chosen to be applied on the full dataset.

8.5.2.4   Training Parameters

Table 42 shows the training parameters for the LSTM models. To avoid overfitting, EarlyStopping and ReduceLROnPlateau was implemented as the callbacks function.

Training parameter	LSTM	LSTM with Class Weights
Epoch	17	52
Learning Rate	Adam Optimiser: 1e-5	Adam Optimiser: 1e-5

Table 42. Training Parameters of LSTM models

8.5.3 Bidirectional Encoder Representations from Transformers

8.5.3.1   BERT

BERT (Bidirectional Encoder Representations from Transformers) [2] is a pre-trained deep learning model used for natural language processing tasks. BERT's bidirectional training on the Transformer attention network allows BERT to have the advantage against single directional models in terms of understanding contexts. As a pre-trained model, BERT has two objectives, masked language modeling and next sentence prediction.

1. For masked language modeling, a certain percentage of words in each sentence are randomly masked and the BERT model is trained to predict the masked words based on the surrounding context.

2. Next Sentence Prediction, the BERT model is trained to predict whether two given sentences are contiguous or not

The fine-tuning of the BERT model for the sentiment analysis task is initiated with a classification head with a dropout layer of probability 0.5 and a linear layer of size input 768. The BertTokenizer uses the WordPiece algorithm to tokenize words and a truncation with a max length of 256. The BertTokenizer returns a set of input_ids and attention masks. Inputs_ids are a sequence of numeric IDs that represent the input

texts. Each ID corresponds to a specific token in the input text. Attention masks are a binary sequence that indicates which tokens in the input_ids sequence should be attended to by the BERT model, the attention mask is used to mask out the padded tokens so they do not contribute to the final output of the BERT model.

8.5.3.2 RoBERTa

RoBERTa (Robustly optimized BERT Pre-Training Approach) [3] is a transformer-based language model developed by FaceBook. The key difference between RoBERTa and BERT is the training technique used. RoBERTa uses dynamic masking during training, where the masking pattern is randomly selected for each example. This benefits the model to learning more effectively by preventing it from memorizing the position of tokens in the input. In addition, RoBERTa removes the next sentence prediction task used in BERT. Similar to BERT, the RobertaTokenizer returns a set of input_ids and attention masks and the fine-tuning of RoBERTa is initiated with a classification head with a dropout layer of probability 0.5 and a linear layer of size input 768.

8.5.3.3 Evaluation

	BERT	RoBERTa
Training Time	141.07s	146.74s
Accuracy	0.85	0.86
Precision (neg)	0.67	0.62
Precision (pos)	0.88	0.91
Precision (neu)	0.80	0.78
Precision	0.78	0.77
Recall (neg)	0.48	0.86
Recall (pos)	0.94	0.91
Recall (neu)	0.70	0.73

Recall	0.70	0.83
F1-score (neg)	0.56	0.72
F1-score (pos)	0.91	0.91
F1-score (neu)	0.74	0.75
F1-score	0.74	0.79

Table 43. Classification Report of BERT Models

From the results obtained above in Table 43, it is observed that the RoBERTa model performs better than the BERT model. This suggests the effectiveness of RoBERTa's improved pre-training methods and use of more diverse data, including dynamic masking, and more robust training objectives. Thus, the RoBERTa classification model will thus be chosen to be applied on the full dataset.

8.5.3.4   Training Parameters

Table 44 shows the training parameters for the Transformer models. Under 5 epochs, the model was able to achieve greater than 80% validation accuracy before plateauing.

Training parameter	BERT	RoBERTa
Epoch	5	5
Learning Rate	Adam Optimiser: 1e-5	Adam Optimiser: 1e-5

Table 44. Training Parameters of Transformer models

8.6 Model Selection

A subset of the classification report is shown in Table 45 below.

	Training Time	Accuracy	F1-score (neg)	F1-score (pos)	F1-score (neu)	F1-score
Ensemble	-	0.82	0.64	0.89	0.64	0.72

LSTM	38.70s	0.81	0.54	0.88	0.67	0.70
roBERTa	146.74s	0.86	0.72	0.91	0.75	0.79

Table 45. Classification Results of All Models

From the results obtained above in Table 45, it is observed that the RoBERTa model performs better than all the other classification models. Thus, it will be the final chosen classification model to be applied on the full dataset.

8.6.1 Application on Full Dataset

Figures 27 and 28 below show the counts of each sentiment in the full dataset as identified by the selected classification model.

```
pos     14252
neu      4488
neg      1555
```

Figure 27. Sentiment Counts

Figure 28. Sentiment Counts Plot

9   User Interface (UI)

9.1   <mark>Features</mark>

9.1.1 Search Engine Page

Upon launching the Crypto.io web application, users will be redirected to the search engine page. In the search bar, users can enter their own freeform query and thereafter, click the 'Search' button. The search engine will then use the query to retrieve the top K ranked relevant tweets. Note that ranking is done by a score that the index gives the preprocessed text data. The tweets, as well as information related to them, will be displayed to the user, as shown in Figure 29 below.

Figure 29. UI Search Engine Page

9.1.2 Search Filters

The slider on the `Number of search results` in the `Search filters` section of the page allows users to specify the number of tweets they want to retrieve. In addition, users can narrow down the search results according to the NFTs they want by clicking the dropdown list on the `Specific NFT(s)`. An example is shown in Figure 30 below where the top 3 relevant `Bored Ape Yacht Club` and `CloneX` tweets will be retrieved and displayed to the user.
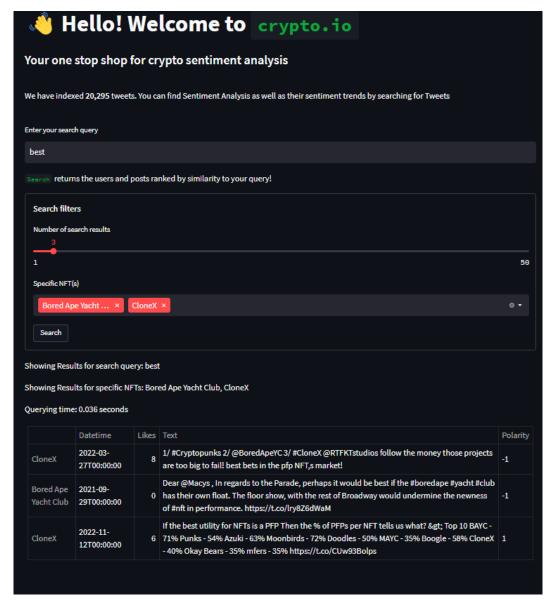

Figure 30. UI Multifaceted Search

9.1.3 Navigation

Users can navigate between different pages in the web application using the sidebar. Here, the `Search Engine Page` is represented by `cypto.io`, the `Classification

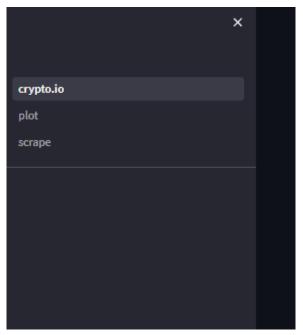Graphs Page` is represented by `plot` and the `Twitter Scraper Page` is represented by `plot`.

Figure 31. Navigation

9.1.4 Classification Graphs Page

Upon navigating to the Classification Graphs Page, users can select different NFTs by clicking the `Specific NFT Graphs` dropdown list in the `Search filters` section of the page. Upon selection, the release date of the NFT, a sentiment count plot, as well as a time series graph will be displayed to the user. The user can use this information to make a more informed decision on whether they want to purchase the NFT.

Figure 32. UI Classification Graphs Page

9.1.4.1   Sentiment Counts Plot

The count plot displays the number of tweets for each sentiment for that specific NFT in the form of a bar graph. The legends used are -1 for negative sentiment (red), 0 for neutral sentiment (blue) and +1 for positive sentiment (green). An example is shown in Figure 33 below.
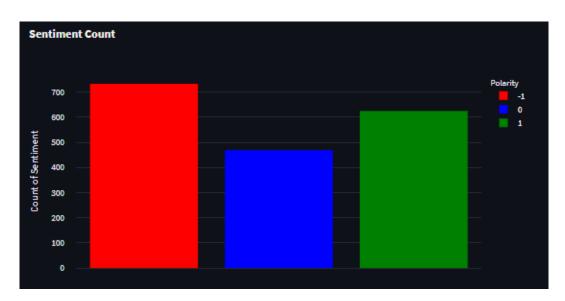

Figure 33. UI Sentiment Count Plot

9.1.4.2   Time Series Graph

The time series graph displays the average sentiment score across all the tweets for that NFT in that specific time period. The scores are calculated by aggregating sentiment scores for tweets retrieved monthly. An example is shown in Figure 34 below.
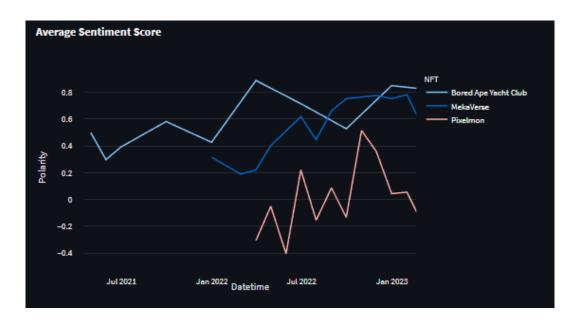

Figure 34. UI Average Sentiment Score Plot

9.1.5 Twitter Scraper Page

Upon navigating to the Twitter Scraper Page, users can scrape new twitter data according to filters they specify. They can either search by keyword or query, specify the time period they would like to collect data from, as well as the number of tweets to scrape. This is shown in Figure 35 below. After data is scraped, they will be preprocessed, classified and indexed. Clicking the `Display Tweets` button displays them as a dataframe to the user. In addition, users can save this information either as a CSV or JSON file by clicking on the `Download as CSV` or `Download as JSON` button respectively. This is shown in Figure 36 below.
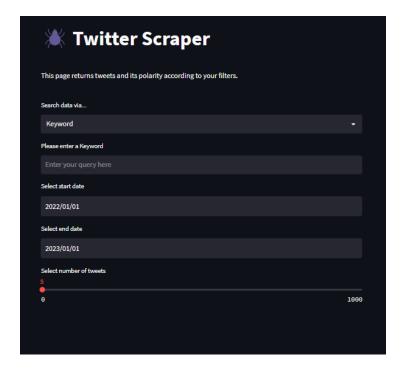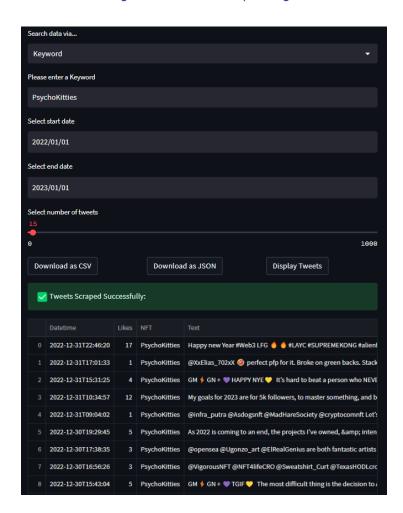
Figure 35. Twitter Scraper Page

Figure 36. Scraping Example

10 Conclusion

In conclusion, this project showed that there is a wealth of information available on the topic of NFTs, which reflects the growing public interest, as well as the range of emotions and opinions associated with this emerging digital asset, expressed across various platforms, including social media (Twitter).

This project also revealed some challenges associated with information retrieval and sentiment analysis related to NFTs, such as the lack of standardized terminology, the rapid pace of change in the market, as well as the need to accurately classify the tone of the language used. It is essential to take these challenges into account to ensure that the information retrieved and the sentiment analysis is accurate and reliable.

This project highlights the importance of information retrieval and sentiment analysis in understanding the complex and dynamic world of NFTs. The various analyses done can help to develop more informed and nuanced perspectives on NFTs, ensuring the long-term viability and sustainability of purchasing them.

11 Submission Links

1. YouTube link to video presentation:

https://youtu.be/4MrnHcFe5SA

2. Dropbox link to compressed file with crawled text data, queries and their results, evaluation dataset, automatic classification results, and any other data:

https://www.dropbox.com/scl/fo/lcod9n8acxtytoaqsqkih/h?dl=0&rlkey=tvg5pdzqggelpiv44jjw5thlr

3. Dropbox link to compressed file with all source codes and libraries:

https://www.dropbox.com/scl/fo/rxg2tekzxcnkub5zlvk2i/h?dl=0&rlkey=lnp79epkzhvlnpzad4o4kmzqs

4. A github repository with all source codes and libraries, with a readme file that explains how to compile and run the source codes:

https://github.com/jolenetan07/CZ4034-IR-Group10.git

12 References

Abueg, Ralf. "Elasticsearch: What it is, How it works, and what it's used for -." *Knowi*,

    7 March 2020, https://www.knowi.com/blog/what-is-elastic-search/. Accessed

    4 April 2023.

ArcGIS Pro. "How LightGBM algorithm works—ArcGIS Pro | Documentation."

    ArcGIS Pro Resources | Tutorials, Documentation, Videos & More,

    1995-2022,

    https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-lightgbm-wo

    rks.htm. Accessed 4 April 2023.

Awan, Abid Ali. "A Complete Guide to Data Augmentation." *DataCamp*, November

    2022, https://www.datacamp.com/tutorial/complete-guide-data-augmentation.

    Accessed 4 April 2023.

Barthere, Aurelie. "Navigating Crypto's Peaks and Troughs: Can We Have A

    Sentiment Indicator for NFTs?" *Nansen*, 1 July 2022,

    https://www.nansen.ai/research/a-sentiment-indicator-for-nfts. Accessed 4

    April 2023.

Chauhan, Nagesh Singh. "Decision Tree Algorithm, Explained." *KDnuggets*, 9

    February 2022,

    https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

    Accessed 4 April 2023.

Devlin, Jacob, et al. "[1810.04805] BERT: Pre-training of Deep Bidirectional

    Transformers for Language Understanding." *arXiv*, 11 October 2018,

    https://arxiv.org/abs/1810.04805. Accessed 4 April 2023.

Elasticsearch. "What is Elasticsearch?" *Elastic*, 2023,

    https://www.elastic.co/what-is/elasticsearch. Accessed 4 April 2023.

Golomb, Michael. "Rise Of A New Disruptor: How NFTs Are Revolutionizing The Art

And Entertainment Worlds." *Forbes*, 7 September 2021,

https://www.forbes.com/sites/forbesbusinesscouncil/2021/09/07/rise-of-a-new-

disruptor-how-nfts-are-revolutionizing-the-art-and-entertainment-worlds/?sh=6

505a47e1a90. Accessed 4 April 2023.

Hoehne, Joshua, and step guide. "How to Scrape Tweets With snscrape | by Martin

Beck." *Better Programming*, 4 December 2020,

https://betterprogramming.pub/how-to-scrape-tweets-with-snscrape-90124ed0

06af. Accessed 4 April 2023.

Hugging Face. "RoBERTa." *Hugging Face*, 2022,

https://huggingface.co/docs/transformers/model_doc/roberta. Accessed 4

April 2023.

IBM. "What is Naïve Bayes." *IBM*, 2020, https://www.ibm.com/topics/naive-bayes.

Accessed 4 April 2023.

Iconiq Inc. "BERT NLP Model Explained for Complete Beginners." *ProjectPro*, 2

February 2023, https://www.projectpro.io/article/bert-nlp-model-explained/558.

Accessed 4 April 2023.

Joby, Amal. "What Is K-Nearest Neighbor? An ML Algorithm to Classify Data." *Learn

G2*, 19 July 2021, https://learn.g2.com/k-nearest-neighbor. Accessed 4 April

2023.

Karani, Dhruvil. "Introduction to Word Embedding and Word2Vec | by Dhruvil

Karani." *Towards Data Science*, 1 September 2018,

https://towardsdatascience.com/introduction-to-word-embedding-and-word2ve

c-652d0c2060fa. Accessed 4 April 2023.

kingsley, Ibekwe. "How to Scrape Millions of Tweets using SNSCRAPER." *Medium*,

 24 May 2022,

 https://medium.com/machine-learning-mastery/how-to-scrape-millions-of-twee

 ts-using-snscraper-aa47cee400ec. Accessed 4 April 2023.

Leray, Loic. "Dealing with Imbalanced Data in TensorFlow: Class Weights." *Towards*

 Data Science, 8 June 2021,

 https://towardsdatascience.com/dealing-with-imbalanced-data-in-tensorflow-cl

 ass-weights-60f876911f99. Accessed 4 April 2023.

Liu, Yinhan, et al. "[1907.11692] RoBERTa: A Robustly Optimized BERT Pretraining

 Approach." *arXiv*, 26 July 2019, https://arxiv.org/abs/1907.11692. Accessed 4

 April 2023.

Microsoft Corporation. "Parameters Tuning — LightGBM 3.3.5.99 documentation."

 LightGBM's documentation!, 2022,

 https://lightgbm.readthedocs.io/en/v3.3.2/Parameters-Tuning.html. Accessed

 4 April 2023.

NVIDIA. "XGBoost – What Is It and Why Does It Matter?" *NVIDIA*, 2023,

 https://www.nvidia.com/en-us/glossary/data-science/xgboost/. Accessed 4

 April 2023.

Ogunbiyi, Ibrahim Abayomi, and Martin Beck. "Web Scraping with Python – How to

 Scrape Data from Twitter using Tweepy and Snscrape." *freeCodeCamp*, 12

 July 2022, https://www.freecodecamp.org/news/python-web-scraping-tutorial/.

 Accessed 4 April 2023.

Pykes, Kurtis. "Oversampling and Undersampling. A technique for Imbalanced… | by

 Kurtis Pykes." *Towards Data Science*, 10 September 2020,

https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56

dcf. Accessed 4 April 2023.

scikit-learn developers. "1.9. Naive Bayes — scikit-learn 1.2.2 documentation."

Scikit-learn, 2007-2023,

https://scikit-learn.org/stable/modules/naive_bayes.html. Accessed 4 April

2023.

scikit-learn developers. "1.10. Decision Trees — scikit-learn 1.2.2 documentation."

Scikit-learn, 2007-2023, https://scikit-learn.org/stable/modules/tree.html.

Accessed 4 April 2023.

scikit-learn developers. "sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.2.2

documentation." *Scikit-learn*, 2007-2023,

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighbor

sClassifier.html. Accessed 4 April 2023.

scikit-learn developers. "sklearn.svm.SVC — scikit-learn 1.2.2 documentation."

Scikit-learn, 2007-2023,

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

Accessed 4 April 2023.

Singh, Taranjeet. "How to Measure the Efficacy of Your Sentiment Analysis Model."

SearchUnify, 2 December 2022,

https://www.searchunify.com/sudo-technical-blogs/how-to-measure-the-efficac

y-of-your-sentiment-analysis-model/. Accessed 4 April 2023.

Smith, John. "Query string query | Elasticsearch Guide [8.7]." *Elastic*, 2023,

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-quer

y-string-query.html. Accessed 4 April 2023.

Smith, Lisa. "Using Query String Queries in Elasticsearch." *Compose*, 24 May 2016,

> https://www.compose.com/articles/using-query-string-queries-in-elasticsearch/

> . Accessed 4 April 2023.

Stecanella, Bruno. "Support Vector Machines (SVM) Algorithm Explained."

> *MonkeyLearn*, 22 June 2017,

> https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/.

> Accessed 4 April 2023.

Stecanella, Bruno. "Understanding TF-ID: A Simple Introduction." *MonkeyLearn*, 10

> May 2019, https://monkeylearn.com/blog/what-is-tf-idf/. Accessed 4 April

> 2023.

THREEKIT INC. "The Rise and Rise of NFTs." *Threekit*, 2022,

> https://www.threekit.com/the-rise-and-rise-of-nfts. Accessed 4 April 2023.

Verma, Yugesh. "A Complete Guide to LSTM Architecture and its Use in Text

> Classification." *Analytics India Magazine*, 10 September 2021,

> https://analyticsindiamag.com/a-complete-guide-to-lstm-architecture-and-its-u

> se-in-text-classification/. Accessed 4 April 2023.

xgboost developers. "XGBoost Parameters — xgboost 1.7.5 documentation."

> *XGBoost Documentation*, 2022,

> https://xgboost.readthedocs.io/en/stable/parameter.html. Accessed 4 April

> 2023.